Estimating the Reputation of Newcomer Web Services Using a Regression-Based Method

Okba Tibermacine, Chouki Tibermacine, Foudil Cherif

Estimating the Reputation of Newcomer Web Services Using a Regression-Based Method

Okba Tibermacine[a], Chouki Tibermacine[b], Foudil Cherif[a]

[a]*Biskra University, LESIA, P.B. 145 R.P, Biskra 07000, Algeria*
[b]*LIRMM, CNRS and Montpellier University, France*

Abstract

In this paper, we propose a novel method to estimate the initial reputation values of newcomer web services. In fact, the reputation of web services is one of the criteria used for recommending services in service-oriented computing environments. The lack of evaluating the initial reputation values can subvert the performance of a service recommendation system making it vulnerable to different threats like whitewashing and Sybil attacks, which negatively affect its quality of recommendation. The proposed method uses Quality of Service (QoS) attributes from a side, and reputation values of similar services from the second side, to estimate the reputation values of newcomer services. Basically, it employs regression models, including Support Vector Regression, in the estimation process of the unknown reputation values of newcomers from their known QoS values. We demonstrate the efficiency of the method in estimating the reputation of newcomer services through statistical evidences gathered from experimentations conducted on a set of real-world web services.

Keywords: Web services recommendation, reputation measurement, support vector regression, feedback rating, honest and malicious service raters.

Email addresses: o.tibermacine@univ-biskra.dz (Okba Tibermacine), Chouki.Tibermacine@lirmm.fr (Chouki Tibermacine), foud_cherif@yahoo.fr (Foudil Cherif)

1. Introduction

Web Service recommendation systems (WSRSs) provide a precious assistance to users in selecting the best available Web Services (WS) to implement their business processes. To recommend services, WSRSs manage different kinds of metrics related to services, among which **reputation**. This subjective quality metric is an aggregation of feedback ratings gathered from service users. It reflects how a given service is perceived by its users which is a good indicator about its Quality of Experience (QoE). The management of reputation plays a significant role in such systems. Recently, many reputation management models have been proposed to accurately evaluate the reputation of web services [1–8]. Although these models have addressed many aspects in reputation evaluation such as user credibility, time sensitivity, personalized preferences, majority ratings, The evaluation of **newcomer** Web services with the absence of feedback ratings is still an important aspect that it has not been tackled thoroughly.

Indeed, assigning reputation values to newcomer (newly published and/or never used) services that have an empty rating history is an important and challenging issue due to the following reasons:

- WSRSs have to assign fair reputation values to newcomer services to enhance their visibility to users and to give these services a chance to compete with longstanding similar services during service selection phases. Hence, it provides a solution to the "cold start" problem, which describes the situation in which a recommender system is unable to make a meaningful recommendation due to an initial lack of users feedback-ratings [12].

- Assigned reputation values attributed by a WSRS have to reflect the non-functional characteristics (QoS values) of a particular service, and should not be general values related to the state of the recommender system itself, such as assigning the average reputation value assessed in the system to all newcomer services [13], or assigning a fixed reputation value based on the rate of maliciousness in the system such as proposed in [14] as an elegant solution to the aforementioned problem.

- The lack of a correct estimation of initial reputation values of newcomer services may subvert the performance of the WSRS itself, making it vulnerable to different threats [11] (e.g., the Sybil attack [15]).

- WSRSs have to provide a solution to the *Whitewashing* problem too [9]. Whitewashing (i.e. changing the identity of a malicious user/service in the system) occurs when an entity leaves the recommendation system, then it reintegrates it with a new identity in order to erase its poor reputation acquired with its previous identity [14].

Though some notable solutions have been proposed in the literature for evaluating the reputation of newcomer services (e.g. [13, 14, 16–18]), most of these solutions assign the same initial reputation value to every newcomer service. Or, they do not offer a complete solution that addresses all the previous challenges.

In this paper, we propose a new model that refines and completes our initial proposition [19] to estimate initial reputation values of newcomer services in WSRSs. A correct estimation of these values allows better recommendations, thus a better help in selecting web services that satisfy clients' requirements.

Even though reputation is a subjective measure, it reflects users' satisfaction about a service's offered functionality and Quality of Service. It has been observed that fair feedback ratings provided by the majority of honest users are correlated, even with a slight deviation (due to differences between raters' opinions), with the QoS of used services [20]. Hence, we use QoS and reputation data of longstanding services to build a reputation estimation model for bootstrapping the reputation of newcomer services. We mean by "long-standing services" the services that have long feedback records constructed from collected user feedback ratings.

In fact, QoS and reputation values could have a linear or nonlinear correlations. Thus, we employ in our propositions both (i) a Linear Regression Model as a reputation estimator to deal with linear relationships, and (ii) a Support Vector Regression (SVR) model as a general reputation estimator to deal with general cases (that is, cases with linear or nonlinear relationships between reputation values and QoS).

Basically, our reputation estimation method is built on three main phases:

1. **Provider reputation evaluation:** The system assesses the reputation of the new service's provider from reputation values of its previously published web services. Then, provider's reputation is employed in bootstrapping the reputation of the newcomer service.
2. **Similarity-based estimation of newcomer's reputation:** In this phase, the system selects among long-standing web services those which

are functionally similar to the newcomer service. Then, the system selects top-K neighbor services that have high QoS value correlation. Finally, the system evaluates the reputation of the newcomer service based on its neighbors' reputation values (Technique 1), or using a multiple regression model (Technique 2) that is built from i) reputation values and ii) QoS values of the service's similar neighbors.

3. **Support Vector Regression-based estimation of newcomers reputation:** In this phase, the system deals with general cases (e.g. new services published by a new provider with no similar long-standing services in the system). The system uses a Support Vector Regression (SVR-) model [21, 22] which is largely employed for forecasting data in both linear and nonlinear problems. The system trains the SVR-model using the normalized QoS and reputation values of all services in the system. Initial QoS values of newcomer services are then used by the SVR-model to estimate their initial reputation values.

The main goal of this work is two-fold:

– First, it provides a solution to evaluate the reputation of newcomer web services that correctly reflects their quality. This enhances their visibility to end users and improve the overall quality of the recommendation system, by providing a solution to the cold-start problem.

– Second, it provides a solution to the whitewashing problem by looking to functionally similar services (including services that have left the system) which are recorded in the system registries.

We evaluated the proposed reputation estimation method on a set of real-world web service. We compared the obtained results with competing approaches from the state of the art.

The remaining of the paper is organized as follows. In Section 2, we present the reputation estimation algorithm. Phases one, two and three of the method are detailed respectively in Sections 3, 4, and 5. We show the results of our experiments in Section 6. We discuss the related work in Section 7, before concluding the paper in Section 8.

2. Reputation estimation method

We suppose that the estimation model is implemented by a reputation management module of a web service recommendation system. This module

Input: S_i // Newcomer service
Output: $\hat{R}_i$ // Estimated Reputation
Begin

1: $\lambda = 0.3$; TopkBool = false;
2: **if** (Provider(S_i) $\in$ *ProviderList*) **then**
3: prReputation = providerReputation(Provider(S_i)) ; // compute providers reputation
4: simServiceSet = getSimilarServices(S_i,*ServiceList*); // get functionnaly similar services
5: **if** (simServiceSet$\neq \emptyset$) **then**
6: **for all** ($S_j \in$ simServiceSet) **do**
7: QoSim [S_j.index] = PCC(normalize(Qos(S_i)),normalize(Qos(S_j))); // using Eq. 4
8: **if** (QoSim [S_j.index] > 0) **then**
9: TopKset.add(S_j) ; // TopK neighbors selection
10: **end if**
11: **end for**
12: **if** (TopKset ==$\emptyset$) **then**
13: TopkBool = true ; // To continue from line 33
14: **else**
15: R_{Min} = MinReputation(TopKset) ;
16: R_{Max} = MaxReputation(TopKset) ;
17: **if** ($R_{Max} - R_{Min} < \lambda$) **then**
18: num = denom = 0 ; // Technique 1, Using Eq. 6
19: **for all** ($S_j \in$ TopKset) **do**
20: num += QoSim [S_j.index] $\times$ getReputation(S_j) ;
21: denom += QoSim [S_j.index] ;
22: **end for**
23: $\hat{R}_i = \frac{num}{denom}$;
24: **else**
25: MLRmodel =buildMLRegressionModel(topKset) ; //Tech. 2 - linear regression
26: $\hat{R}_i$= estimateReputation(Qos(S_i),MLRmodel) ;
27: **end if**
28: **end if**
29: **else**
30: $\hat{R}_i$= prReputation ; // Assign provider reputation
31: **end if**
32: **end if**
33: **if** (!(Provider(S_i) $\in$ *ProviderList*) || TopkBool) **then**
34: simServiceSet = getSimilarServices(S_i,*ServiceList*);
35: **if** (simServiceSet$\neq \emptyset$) **then**
36: SimVector = Similarities(S_i, simServiceSet);
37: aService =HighestScoreService(SimVector);
38: **if** (Max(simVector)==1 && hasLeft(aService)) **then**
39: $\hat{R}_i$ =getReputation(aService);
40: **else**
41: MLRmodel =buildMLRegressionModel(simServiceSet) ; //Apply tech.2 for SimServSet
42: $\hat{R}_i$= estimateReputation(Qos(S_i),MLRmodel) ;
43: **end if**
44: **else**
45: SVRModel = BuildSVRModel(serviceList); // Build Support Vector Regression model
46: $\hat{R}_i$ = estimateSVRReputation(Qos(S_i),SVRModel);
47: **end if**
48: **end if**
End

Algorithm 1: Reputation estimation algorithm

enables the collection of feedback ratings from service users, the assessment of reputation values of web services, and the monitoring and/or the collection of services' QoS data. Moreover, the system indexes Web service descriptions and keeps information of all services, including services that left the system.

When a newcomer Web service S_i arrives to the system, we assume that it comes with an initial QoS vector $Q_{S_i}^{init} =< q_{i,1}, q_{i,2}, ..., q_{i,k} >$. These QoS values ($Q_{i,j}$) are provided during registration time by the service provider ($Pr(S_i)$) as advertised QoS data. This data can also be updated by the system after a period of service monitoring and testing (for that propose, the system can use one of the approaches proposed in this survey [23].)

Algorithm 1 presents the process that covers three phases to estimate the reputation of a newcomer web service S_i. First, the system checks whether the service provider is known by the system, that is, the service provider belongs to the list of providers *ProviderList* that have published services in the system. In the positive case, the system calculates the reputation of this provider, denoted *prReputation*, based on the reputation values of its long-standing services (Line 3 in Algorithm 1). We present the details on how to calculate the reputation of the provider in Section 3.

Afterwards, the system seeks for long-standing services that provide functionalities which are similar to those provided by the newcomer (Line 4). To evaluate the functional similarity between web services, we use the approach proposed in [24]. If *simServiceSet*, which denotes the set of similar service, is not empty, the system selects, using positive Pearson Correlation Coefficient (PCC) values, a subset of top-K neighbors with close QoS vectors to the newcomer's QoS vector (Lines 6-11). Both PCC and top-K are widely and effectively used in recommender systems for the selection of similar elements.

When the maximum distance between reputation values of neighbor services in Top-K is relatively small (less then $\lambda = 0.3$ for instance), which means all reputation values of neighbors are close to each other, the system estimates the reputation of the newcomer service as the mean weighted reputation values of its neighbors, where weights are their PCC values (Lines 15-23). Otherwise, the system builds a multiple regression model (see Section 4.4.2 for details) using QoS vectors and reputation values of top-K neighbors, and therefore estimates the reputation value of S_i using this model (Lines 25-26).

In the case where *simServiceSet* is empty then the system assigns to the reputation of S_i the reputation values of its provider "prProvider" (Line 30). This is motivated by the fact that if the provider has a good reputation, it is likely that its new web service will have a good starting reputation too.

Besides, when the provider of the service is also new to the system, we check if it is a whitewashing situation (lines 33-38 in Algorithm 1). The system retrieves all similar long-standing services and compares their similarity scores with the newcomer service (similarity scores range between 0 and 1, where 1 means that services are totally similar and 0 otherwise). If the highest similarity score equates to one, and the similar service has left the system, then, the provider of the newcomer service becomes suspicious, and we assign the (old) reputation value of the left service to the reputation values of the newcomer service (line 38 in the algorithm). Otherwise, we use Technique 2, where the system builds a multiple linear regression model from QoS and reputation values of similar services ($simServiceSet$). Then, the system estimates the reputation of S_i using the built model (Lines 41-42).

When the newcomer service and its provider are both new, and there are no similar services in the system, we go to Phase 3 (detailed in Section 5). The system builds a Support Vector Regression model from QoS vectors and reputation values of all long-standing Web services in the system (line 44 in the algorithm). Similarly to Phase 2, the model gives also an estimation of service reputation based on the service's initial QoS. The estimated value is assigned to the reputation of the newcomer service S_i.

3. Provider reputation

The reputation of a provider mainly depends on the quality of its offered services, thus on their reputation scores. In this phase, we calculate the reputation of a given provider as the weighted arithmetic mean of reputation scores of its offered services. Given a provider Pr_x, let $Services(Pr_x) = \{S_i\}, i = 1, ..., n$ denote the set of n web services provided by Pr_x, and $\Omega(S_i)$ be the number of users who rated service S_i. The reputation of a provider Pr_x is calculated as follows:

$$RP(Pr_x) = \begin{cases} \frac{(\sum_{i=1}^{n} \Omega(S_i) * R_i)}{\sum_{i=1}^{n} \Omega(S_i)} & if \quad Services(Pr_x) \neq \emptyset \\ 0 & Otherwise \end{cases} \tag{1}$$

where,

- R_i is the reputation of service S_i that belongs to the provider's service set ($S_i \in services(Pr_x)$).
- $\emptyset$ denotes the empty set.

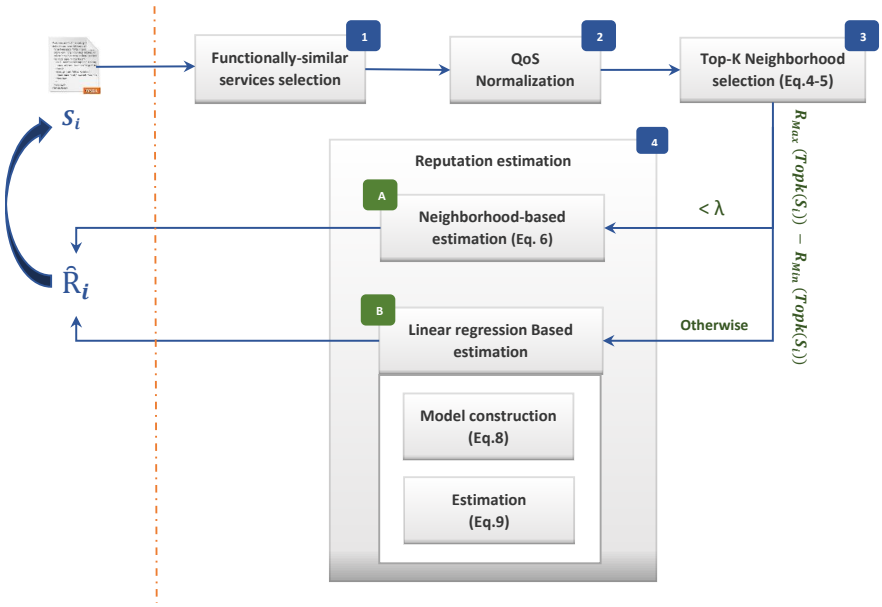

Figure 1: The second phase for estimating the reputation of a newcomer service S_i.

The reputation of a new provider, in case of introducing its new service, will be set to the estimated reputation of its newcomer service which is calculated using one of the following estimation phases.

4. Reputation estimation from similar services

Since users rate functionally-similar web services (i.e. services that provides same functionalities) based on the same criteria, we consider that it is possible to estimate the reputation of newcomer web services using reputation scores of its similar services, which are calculated by aggregating users' ratings. In this phase, a four-step technique is proposed to estimate this reputation based on QoS values of the newcomer service S_i and the existing long-standing similar services (see Figure 1). As mentioned above, the initial QoS values of S_i are represented by the QoS vector $Q_{S_i}^{init} =< q_{i,1}, q_{i,2}, ..., q_{i,k} >$.

The method for estimating the initial reputation of S_i denoted $\hat{R}_i$ is detailed in the following sub-sections.

4.1. Step 1 - Functionally-similar service selection

First, the system selects from its databases long-standing services that offer the same functionalities (e.g. Weather forecasting services, currency services, transportation services, etc.) to the service S_i. In the literature many approaches that compute the similarity between web services are proposed (e.g. [25–27]). In this work, we choose to use the approach proposed in [27] to calculate the similarity between a newcomer web services and the existing services. The approach assesses the similarity between two web services by comparing their WSDL definitions using several lexical and semantic metrics. The results of the similarity assessment are scores that range from 0 and 1, where 0 represents a total dissimilarity and 1 a total similarity. The compared web services with a similarity score greater or equal than a fixed threshold (e.g., 0.75) are considered as similar. The result of this step is a set of similar services denoted by $simServiceSet$.

4.2. Step 2 - QoS normalization

Second, the system retrieves and normalizes QoS and reputation values of each service in the $simServiceSet$. Let $simServiceSet = \{S_j\}, (j = 1, .., m)$ be the set of m similar services to the newcomer service S_i. Each similar service S_j in this set has a QoS vector $Q_{S_j} =< q_{j,1}, q_{j,2}, ..., q_{j,k} >$ and a reputation value R_j calculated by the system from user feedback ratings. Besides, the newcomer service S_i is defined by the vector $Q_{S_i}^{init}$ that represents its initial known QoS values, and $\hat{R}_i$ that represents the unknown reputation value (to be estimated).

Afterwards, the system normalizes all QoS values in the range $[0, 1]$.

Thus, each QoS value, $QosVal \in \{q_{j,l}, (j = 1, .., m; \quad l = 1, .., k)\}$ is replaced in its vector by its normalized value $NewQosVal$ which is calculated as follows:

$$NewQosVal = \frac{QosVal - MinVal}{MaxVal - MinVal} \tag{2}$$

Where $MinVal$ and $MaxVal$ are respectively the minimum and maximum recorded values in the system for that QoS metric (with the index l). Note that some of QoS metrics have values that are interpreted inversely, i.e. the higher is the value, the lower is the quality. This includes execution time and price. Thus, the scaled value $NewQosVal$ is calculated as follows:

$$NewQosVal = 1 - (\frac{QosVal - MinVal}{MaxVal - MinVal}) \tag{3}$$

4.3. Step 3 - QoS-similar neighborhood selection

The more the QoS values of the newcomer services are close to the QoS values of other services, the more its reputation value is close to their reputation values. Thus, the system in the third step selects the QoS-Similar neighbors from the *simServiceSet* by calculating similarities between the QoS Vectors of Web services. These similarities could be calculated using PCC (Pearson Correlation Coefficient) or VSS (Vector Space Model Similarity) estimates, which are used in recommendation systems [28–30]. PCC can generally achieve higher performance than VSS [31]. Therefore, we employ PCC for the similarity computation between normalized QoS vectors $Q_{S_j}, (j = 1..m)$ and $Q_{S_i}^{init}$, the QoS vector of the newcomer S_i, using the following equation:

$$PCC(S_i, S_j) = \frac{\sum_{l=1}^{k}(q_{i,l} - \overline{Q_i})(q_{j,l} - \overline{Q_j})}{\sqrt{\sum_{l=1}^{k}(q_{i,l} - \overline{Q_i})^2}\sqrt{\sum_{l=1}^{k}(q_{j,l} - \overline{Q_j})^2}} \tag{4}$$

where, $q_{i,l}$ and $q_{j,l}$ are the values of l'th quality in the QoS vectors $Q_{S_i}^{init}$ and Q_{S_j}, and $\overline{Q_i}$, $\overline{Q_j}$ are the average QoS values of the previous vectors respectively.

From Eq. 4, $PCC(S_i, S_j)$ values belong to the interval $[-1, 1]$, where a larger PCC value indicates higher QoS-similarity between services S_i and S_j. After calculating QoS-Similarities between the newcomer service and services in the *simServiceSet*, a set of top-K neighbors is identified based on PCC values. In this work we ignore negative PCC values because negative values could represent a dissimilarity between compared services, which influences greatly the accuracy of the estimation of reputation in next steps. Thus, the top-K neighbor set of the newcomer service S_i is defined as follows:

$$\begin{aligned} TopK(S_i) = \{S_j \quad | \quad & PCC(S_i, S_j) >> 0; \\ & S_j \in simServiceSet\} \end{aligned} \tag{5}$$

where, $PCC(S_i, S_j)$ is computed using Eq. 4. In case $TopK(S_i)$ equates the empty set ($\emptyset$), then the system moves to Phase 3 of the method, and the estimation of the reputation of the newcomer service $\hat{R}_i$ is calculated using the SVR-based model. Otherwise, its reputation is estimated in the next step.

4.4. Step 4 - Reputation estimation

We propose two techniques to estimate the reputation of a newcomer service based on the data about the services in the top-K neighbor set ($TopK(S_i)$). The first consists in calculating the weighted mean of neighbors reputations (Section 4.4.1), and the second is based on the construction of a multiple linear regression model (Section 4.4.2) from QoS vectors and their corresponding reputation values.

The system selects the first technique when the difference between the maximum and the minimum reputation values of services in the Top-k set is less or equal than a threshold λ (e.g. $\lambda = 0.3$).

$$R_{Max}(TopK(S_i)) - R_{Min}(TopK(S_i)) < \lambda$$

which is the case when all service reputation scores in the Top-K set are close each to the other (i.e. the distance between reputation values do not exceed λ). Otherwise, the system selects the second technique to estimate $\hat{R}_i$.

4.4.1. Neighborhood-based estimation

The estimation of $\hat{R}_i$ is calculated using Eq. 6.

$$\hat{R}_i = \frac{\sum_{j \in TopK(S_i)} (PCC(S_i, S_j) * R_j)}{\sum_{j \in TopK(S_i)} PCC(S_i, S_j)} \tag{6}$$

where $TopK(S_i)$ is the set of neighbors that are functionally and qualitatively (based on their QoS values) similar to the newcomer service S_i, and $PCC(S_i, S_j)$ is the similarity between S_i's and S_j's QoS vectors. Using PCC as a weight in Eq. 6 means that reputation scores of services, whose QoS values are highly correlated with the QoS values of the newcomer service, are assigned with higher weights (i.e PCC values close to 1).

4.4.2. Linear regression-based estimation

The second technique to estimate $\hat{R}_i$ is achieved by constructing a multiple regression model, using QoS and reputation values of top-K service neighbors.

Multiple regressions are statistical techniques used for predicting unknown Y values (a dependent variable) corresponding to a set of X values (independent variables). In our study, the multiple regression is expected to give a model that could relate the reputation values of long-standing services to their QoS values, that is, we consider the dependent variable Y to represent

the reputation of services as a function of multiple QoS attributes (independent variables) such as *response time, availability, throughput, latency, price, etc.* Thus, if we have m services in the $TopK(S_i)$ (S_j , $j = 1, 2, ..., m$), and each service S_j has a QoS vector $Q_{S_j} =< q_{j,1}, q_{j,2}, ..., q_{j,k} >$ that holds k QoS metrics, and each service S_j has a reputation value $R(S_j)$ denoted R_j, the relationships between reputation (the dependent variable) and QoS metrics (independent variables) can be expressed by the following equation:

$$\underbrace{\begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,k} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m,1} & q_{m,2} & \cdots & q_{m,k} \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{pmatrix}}_{\beta} + \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{pmatrix}}_{\varepsilon} = \underbrace{\begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_m \end{pmatrix}}_{Y} \tag{7}$$

where :

- X is the design matrix that packs all regressors (predictors) $q_{l,j}, l = 1, ..., m$ and $j = 1, ..., k$.
- β is the regression coefficient vector (called also slop vector).
- ε is the error vector. Error terms $\epsilon_l, i = 1, .., m$ capture all the factors which influence the dependent variable ($R_l, l = 1, ..., m$) other than regressors ($X_{l,j}, l = 1, ..., m$ and $j = 1, ..., k$).

The multiple regression of the model can be simplified to:

$$R_l = \beta_1 q_{l,1} + \beta_2 q_{l,2} + ... + \beta_k q_{l,k} + \epsilon_l, \qquad l = 1, ..., m \tag{8}$$

where,

- R_l is the response (estimated reputation) of the linear combination of the model terms.
- β_l ($l = 1, ..., k$) represents the unknown coefficients.
- ϵ_l is the error term.

After model construction, the system uses solved values of the unknown coefficients (β_l ($l = 1, ..., k$), and the error term (ε), to estimate the reputation of the newcomer service based on its initial QoS vector using Eq 9.

$$\hat{R_i} = \beta_1 q_{i,1} + \beta_2 q_{i,2} + ... + \beta_k q_{i,k} + \epsilon_i. \tag{9}$$

5. SVR-based reputation estimation

In this phase, the system deals with general cases, where it is unable to recognize the service provider and it cannot find similar services. This phase relies on the use of a Support Vector Regression model [21, 22] to estimate the reputation of newcomer services. The system constructs an SVR model using QoS data and reputation values of all long-standing services.

As a learning algorithm based on statistical learning theory [32], SVR has been applied extensively in non-linear regression/prediction problems (e.g. [33–37]). The essence of the application of this algorithm in our context is to map QoS and reputation values of long-standing services into a higher dimensional space via a non-linear mapping and then to do linear regression in this space (i.e. to find the relationship between QoS and reputation values). The main idea in SVR is summarized as follows:

Given a training set $D = (x_i, y_i); i = 1, 2, ..., N$, $y_i \in \mathbb{R}$, $x_i \in \mathbb{R}^N$, where x_i is the ith input in the N-dimension space, and y_i is the output value corresponding to x_i.

Through a mapping function $\Phi(x)$, SVR maps input data x_i to a high dimension feature space $\mathcal{F}$, which can describe the relationships between inputs x_i and outputs y_i. This linear function is formulated as an SVR function (Eq. 10).

$$f(x) = \omega^T \Phi(x) + b \quad , \quad \Phi : \mathbb{R} \to \mathcal{F}, \quad \omega \in \mathcal{F} \tag{10}$$

Note that the dimension of Φ is not specified. The training with the set D consists in estimating the optimal coefficients ω and b which are determined by minimizing a risk function as follows:

$$R_\omega = \frac{1}{2}\|\omega\|^2 + R_{emp} = \frac{1}{2}\|\omega\|^2 + C \times \frac{1}{2}\sum_{i=1}^{N} |y_i - f(x_i)|_\varepsilon \tag{11}$$

where :

- R_{emp} represents the empirical risks;
- $\frac{1}{2}\|\omega\|^2$ and C denote the Euclidean norm and a cost parameter measuring the empirical risks respectively;
- N is the number of samples;

- $|y_i - f(x_i)|_\varepsilon$ is an ϵ-insensitive loss function, which controls deviation and makes the estimation robust.

The ϵ-insensitive loss function is formulated by Eq. 12.

$$|y - f(x)|_\varepsilon = \begin{cases} 0 & , \quad if \quad |y - f(x)| \leq \varepsilon \\ |y - f(x)| & , \quad else \end{cases} \tag{12}$$

From Eqs. 10, 11 and 12, the coefficient ω and b are determined through the support vector regression method by minimizing the objective function $R(\omega)$:

$$\begin{aligned} & R(\omega, \xi_i, \xi_i^*) = \tfrac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{N}(\xi_i, \xi_i^*) \\ & s.t. \begin{cases} y_i - \omega^T \Phi(x) - b \leq \varepsilon + \xi_i^*, & i = 1, 2, ..., N \\ \omega^T \Phi(x) + b - y_i \leq \varepsilon + \xi_i, & i = 1, 2, ..., N \\ \xi_i, \xi_i^* \geq 0, & i = 1, 2, ..., N \end{cases} \end{aligned} \tag{13}$$

where, ξ_i, ξ_i^* are slack variables (relaxation factors) used to represent the stated excess positive and negative deviation. In Eq. 13, $\frac{1}{2}\|\omega\|^2$ makes the regression function more flat and has better generalization ability. The second part of Eq. 13 is used to reduce the error. The C parameter is a constant number $C \geq 0$. It is used to control the degree of "punishment" of samples beyond the error ε. Then we establish the Lagrange's equation:

$$\begin{aligned} L(\omega, \xi_i, \xi_i^*) = & \tfrac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{N}(\xi_i, \xi_i^*) \\ & - \sum_{i=1}^{N} \alpha_i[(\varepsilon + \xi_i) + y_i + \omega^T \Phi(x_i) + b] \\ & - \sum_{i=1}^{N} \alpha_i^*[(\varepsilon + \xi_i^*) + y_i + \omega^T \Phi(x_i) - b] \\ & - \sum_{i=1}^{N}(\lambda_i \xi_i + \lambda_i^* \xi_i^*) \end{aligned} \tag{14}$$

To optimize Eq. 14, the function $L(\omega, \xi_i, \xi_i^*)$ has to be minimal. Then,

we obtain the following dual optimization problem:

$$\begin{aligned} &\min\{\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)[\Phi(x_i).\Phi(x_j)] \\ &\quad+\sum_{i=1}^{N}\alpha_i(\varepsilon-y_i)+\sum_{i=1}^{N}\alpha_i^*(\varepsilon+y_i)\} \\ &s.t.\begin{cases}\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)=0\\ \alpha_i,\alpha_i^*\in[0,C]\end{cases}\end{aligned} \tag{15}$$

The support vector regression problem can be summed up in a quadratic programming problem. By solving the quadratic problem in Eq. 15, the parameter vector ω in Eq. 10 is obtained as follows:

$$\omega=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\Phi(x_i) \tag{16}$$

where α_i and α_i^* are the Lagrangian multipliers which are the solution of $R(\omega,\xi_i,\xi_i^*)$. Thus, the SVR regression function is obtained in the dual space shown below:

$$f(x)=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)k(x,x_i)+b \tag{17}$$

where $k(x,x_i)=\Phi(x).\Phi(x_i)$ is a kernel function. Different SVR models can be elaborated by selecting different kernel functions. One of the widely used kernels is the Gaussian Radial Basis Function (RBF) which performs a nonlinear mapping between the input space and a high dimensional space, and which is easy to implement at the same time [34]. Under the uncertainty that reputation is collinear with QoS data, we choose the RBF Kernel to develop our reputation estimatation SVR model. Thus Eq. 17 is rewritten as follows:

$$f(x)=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\exp\left(\frac{-\|x_i-x\|^2}{2\sigma^2}\right)+b \tag{18}$$

where σ (kernel parameter) is the width of the RBF kernel function. x_i is the input vector of the training data, i.e. the QoS vector of long-standing services in our case. x is the vector of testing data, i.e. the initial QoS vector of the newcomer Web service. The reputation of the newcomer service $\hat{R}_i$ equates to f(x).

Number	Quality	Description	Unit
1	Response time	Time taken to send a request and receive a response	ms
2	Availability	Number of successful invocations / total invocations	%
3	Throughput	Total Number of invocations for a given period of time	invocations/second
4	Successability	Number of response messages / number of request messages	%
5	Reliability	Number of error messages / total number of messages	%
6	Compliance	The extent to which a WSDL document follows WSDL specification	%
7	Best Practices	The extent to which a Web service follows WS-I Basic Profile	%
8	Latency	Time taken for the server to process a given request	ms
9	Documentation	Measure of documentation (i.e. description tags) in WSDL	%

Table 1: QoS metrics selected from QWS dataset

6. Experiments and Evaluation

To evaluate the proposed reputation estimation method, we conducted an experiment on a set of real web services collected from WSDream [38] and QWS [39] datasets. Our experiment has been conducted through three steps, in which we evaluated the three proposed estimation phases. Different accuracy metrics are used to compare the results obtained by our estimation method against results obtained by related state-of-the-art methods.

6.1. Data description and preparation

WSDream dataset holds 5825 web service QoS data evaluated by 339 users in different geographical locations (we have chosen the dataset 2 in WSDream). This dataset holds 339*5825*2 (2 QoS characteristics: response time and throughput). QWS dataset holds 365 web services with 9 QoS characteristics listed in Table 1.

To use a maximum number of QoS metrics with different monitored values of response time and throughput, we selected the services that belong to the two datasets. We matched web services based on their URIs, Names, and WSDL file size. We obtained 409 WSDL files for 356 services, where 53 files from the 409 are redundant WSDL with different endpoint and QoS metrics. Each service in this set has 7 fixed QoS metrics from QWS, and 2 QoS metrics (response time and throughput) that vary based on the observation of 339 users from WSDream. This final web service set is used in the experimentation.

6.2. Evaluation metrics

Statistical accuracy metrics are performance metrics that are used for the evaluation of recommender systems. In this experiment, we use Mean

Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Root-Mean-Squared-Error (RMSE) metrics to measure the quality of the estimation provided by our method in comparison with similar methods.

MAE is a quantity that measures how close are the estimations (predictions) to the eventual outcomes. MAE is defined as follows:

$$MAE = \frac{\sum_{i=1}^{n} |R_i - \hat{R}_i|}{n} \tag{19}$$

MAPE expresses the accuracy as a percentage of the error (e.g. if MAPE is 5, the estimation is off by 5 %). MAPE is defined as follows:

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{R_i - \hat{R}_i}{R_i} \right| \times 100 \tag{20}$$

RMSE is a quantity to measure the difference between predictions and eventual outcomes. RMSE gives a relatively high weight to larger errors. It is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{R}_i - R_i)^2}{n}} \tag{21}$$

In Eqs. 19 and 21, R_i denotes the actual reputation (Reputation which is calculated by aggregating simulated feedback ratings), and $\hat{R}_i$ denotes the estimated (predicted) reputation calculated by the proposed method (or similar methods), and n is the number of tested services.

In addition, we use the correlation coefficient (R) that measures the strength and the direction of a linear relationship between variables (Reputaion and Qos metrics in this case), and the coefficient of determination (R^2) that gives the proportion of the variance of reputation variable that is predictable from QoS metric variables.

6.3. Comparison

To show the efficiency of the proposed method, we compare our Reputation Estimation Method (labeled **REM**) with the following two competing methods (1 and 2) and the three baseline methods (3 to 5):

1. The method based on Malicious user Density (labeled **MDBM**) proposed by Malik et al. [14]: this adaptive bootstrapping approach calculates the initial trust value based on the rate of maliciousness in the

system. It assigns a high initial reputation value when R, the maliciousness rate, is low, and a low reputation value when R is high.

2. "Artificial Neural Network"-based method (labeled **ANN**) proposed by Wu *et al.* [16]. This machine learning method estimates the reputation of newcomer Web services using an artificial neural network model built from Quality of Service values.
3. Minimum Value Method (labeled **MVM**): this approach is used by Zacharia et al. [40], it assigns the minimum possible reputation value to all newcomers.
4. Neutral Method (labeled **NM**) used by Wang et al. [41]. This method assigns the neutral value (0.5) to the reputation of newcomers.
5. Average Reputation Method (labeled **ARM**) used by Huang et al. [13]. This method assigns the average reputation in the system (i.e. the mean reputation value of longstanding services)

6.4. Feedback rating simulation

Due to the current limited availability of feedback rating data, many web service reputation management approaches (e.g. [10, 14, 20]) have used simulation for generating user feedback ratings for assessing service reputation values. Likewise, we have built a Java program that simulates interactions between a set of 409 web services and a set of 339 users.

Each service has an actual performance (overall quality) level (from 1 to 10), denoted *PerfVal*, that represents on a scale of 10 how good is the overall quality provided by the service. PerfVal is calculated based on a utility function (i.e., a single scalar metric to quantify quality perception) of the delivered service, as suggested in [42]. However, in our work, we propose to calculate the utility function with the root mean square, which is a measure of the magnitude of the scaled QoS metrics.

Thus, *PerfVal* of service S_i is assessed as follows:

$$PerfVal(S_i) = 10 \times \sqrt{\frac{\sum_{j=1}^{k} Scal(Q_{i,j})^2}{k}} \tag{22}$$

where, k is the number of used QoS metrics (Q_j, $j = 1, .., k$). And, $Scal(Q_{i,j})$ is the scaling function, which is defined by Eq. 23, if the quality is positive

(i.e., the higher is the value the higher is the quality), and by 1 - the same formula otherwise.

$$Scal(Q_{i,j}) = \frac{Q_{i,j} - Min(Q_j)}{Max(Q_i) - Min(Q_j)} \tag{23}$$

$Min(Q_j)$ and $Max(Q_j)$ are respectively the minimum and maximum recorded values of the quality Q_j.

The program simulates two kinds of users: honest and malicious users. Honest users randomly rate a service based on its *PerfVal* within the interval $[Max(0, PerfVal - 2), Min(PerfVal + 2, 10)]$. For example, if *PerfVal*=7, honest feedback ratings could be 5, 6, 7, 8, and 9. The deviation with ± 2 from *PerfVal* represents the natural variation between user opinions. Malicious users randomly rate the same service outside the previous interval, always on a scale of 10. For instance, if *PerfVal*=7, malicious feedback ratings could be 0, 1, 2, 3, 4 and 10.

In this simulation, we consider different malicious user densities. Whitby et al. [43] and Malik et al. [20] claim that high maliciousness densities are unrealistic in real world applications. Thus we have considered maliciousness densities in the interval [10% - 30 %].

Finally, the reputation of web services is calculated as the mean of the collected feedback ratings. The final reputation values are averages of 10 round-simulation results.

6.5. Provider-based reputation

In this section, we evaluate the accuracy of the reputation estimation of newcomers using the reputation of their providers. We have selected a list of providers that have more than one service in the working dataset. For each provider, the program takes randomly one or more services, depending on the number of its published services and the rate of test, to construct the test-set (i.e. the set of services that are considered as newcomers to the system). The program does the following:

1. Calculate the reputation from simulated feedbacks for all services.
2. Randomly select services to construct the test-set based on a given ratio. This ratio indicates how many services will be considered as a newcomer. Note that each service in the test-set has a calculated reputation.

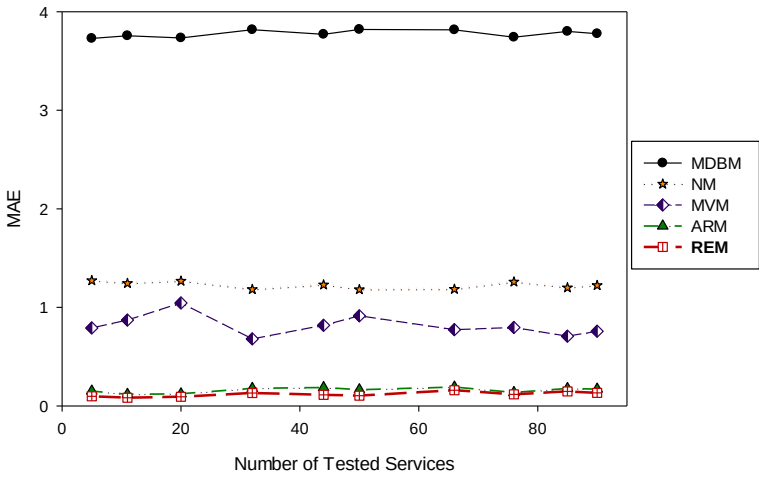

Figure 2: Comparing MEA results between our provider-based estimation and other methods.

3. For each element in the test-set, calculate its provider reputation and assign it to the newcomer's estimated reputation.
4. Evaluate the reputation of the tested service using other methods.
5. For each method, compute the MAE, MAPE and MRSE based on the calculated (from simulated feedbacks) and the estimated reputation (evaluated by the method).
6. Print results.
7. Repeat steps (1-6) for different test-set rates.

The program identified 55 providers that publish more than one service from a total of 251 providers in the working dataset. We have performed several runs, varying the number or tested services (i.e. newcomers). Figure 2 shows the obtained MAE for our method (REM) and other methods. Similar results are obtained with MAPE and RMSE. As we can see, the MAE values given by our method are very close to 0, which means that there is a very slight variation between the calculated reputation from simulation and the estimated reputation from newcomers' providers. In addition, we can see that the assignment of the average reputation in the system (**ARM**) gives also a good result in estimating the reputation of newcomers. However, **REM** still gives better results. It is obvious that the system could not apply this method, especially if the provider of the newcomer service is new too.

Group	Nbr Services	Nbr Tests	MAE	MAPE%	RMSE
Booking	3	1	0.032	0.504	0.045
Calendar	11	4	0.036	0.567	0.056
Currency	7	3	0.162	2.716	0.201
Email	5	1	0.044	0.714	0.087
Financial	3	1	0.067	1.089	0.095
Game	4	1	0.002	0.031	0.003
Geolocation	36	10	0.063	0.970	0.135
Languages	4	1	0.063	1.063	0.108
Login and security	15	5	0.024	0.394	0.042
Lookup	12	4	0.065	1.017	0.136
Math	15	5	0.107	1.850	0.200
News	18	5	0.034	0.539	0.081
Phone and SMS	17	6	0.057	0.901	0.093
Religion	3	1	0.012	0.190	0.017
Trade	10	3	0.044	0.677	0.096
Versioning	4	1	0.015	0.244	0.026
Weather	6	1	0.034	0.538	0.077
MEAN			**0.051**	**0.824**	**0.088**

Table 2: MEA, MAPE and RMSE results using the Neighborhood-based estimation

6.6. Similarity-based reputation

In this section, we present an experiment we have conducted to evaluate the efficiency of Phase 2: similarity-based reputation estimation.

Only services that have similar services are evaluated during this phase. Hence, the program eliminates from the working dataset, all services that have none, one or two similar services. That is, we need at least 3 services (i.e. one for test and two at least to apply Eq. 6) to evaluate one of them using Technique 1 (Section 4.4.1). We need at least 10 similar services (i.e. one service for a test, and at least 9 services data to build the linear regression model with 9 known QoS attributes) to evaluate newcomer reputation using Technique 2.

First, the program analyzed the initial working dataset. It identified 17 groups (clusters) of services with 173 services, as it is shown in Table 2. Elements of each group are functionally similar to each other. Column "Nbr Services" shows the total number of services in the group, and column "Nbr tests" depicts the number of services considered as newcomers.

After several runs, we found that the program, with this working dataset,

always finds Top-K elements for each service, with a small distance between reputation values of these Top-K elements. Thus, it always uses the neighborhood-based estimation technique (1). The obtained MEA, MAPE and RMSE are given in Table 2. We note that these values are the mean values of 10 execution rounds. We can see that the estimated reputations are very close to the actual values with MEA of 0.51 in average. In addition, Technique 1 estimates accurately the initial reputation with a percentage error of 0.824%, which is a good score.

Besides, we applied the "Multiple Linear Regression"-based estimation technique (2) on some groups to test their estimation accuracy. To construct the multiple linear model we need a dataset of at least 9 services. Thus, we limited the use of this technique on groups with a high number of services. Test services were selected randomly, and the construction of the multiple linear model was with 80% of services.

Table 3 provides the obtained MAE, MAPE, RMSE, R and R^2. As we can see, there is a deviation between the estimated reputation values and the calculated values represented by MAE value (0.8464 for example in group "Trade"). This deviation is caused by the size of the trained data. All these groups hold basically a small number of services. However, these values are still close to the calculated values, and the percentage error does not exceed 14% in the worst case. The correlation coefficient (Multiple R) for all groups ranges in [0.78 - 1] which indicates a positive relationship between reputation values and QoS data (where 1 indicates a perfect positive relationship). We see also from R^2 that most of the trained values fit the model (e.g., for Math group 0.87% of the values fit the model). From this experimentation, we can say that the largest the data we use for training, the best the accuracy we obtain during the estimation of newcomer reputation. We conclude that we may safely use this technique too to estimate the reputation of newcomers from their functionally-similar services.

6.7. SVR-based reputation

To construct the Support Vector Regression models, we used the LibSVM API [44] in our Java program. Since we want to construct a model that covers both linear and non-linear relationships between QoS and reputation values in the system, the Gaussian Radial Basis Function (RBF) is chosen, with the kernel parameter $\sigma = 1$, Cost (regularization) parameter C=1, and insensitive coefficient ϵ= 0.00001.

Category	MAE	MAPE(%)	RMSE	R	R2
News	0.4427	7.0962	0.4677	0.9211	0.8484
Currency	0.0755	0.7257	0.0057	0.9820	0.8633
Trade	0.8646	13.6597	0.9120	1.0000	1.0000
Math	0.1815	2.8735	0.2090	0.9364	0.8768
Login and security	0.7195	11.5262	1.1858	0.9152	0.8377
Geolocation	0.0751	1.2063	0.0991	0.7845	0.5741

Table 3: MEA, MAPE, RMSE, R and R^2 results using linear multiple regression

Figure 3: Comparison samples between calculated and estimated reputation with different test-set rates

Maliciousness Rate	**Methods**	Test set = 10%			Test set = 20%			Te
		MAE	**MAPE%**	**RMSE**	**MAE**	**MAPE%**	**RMSE**	**MAE**
10%	MDBM	2.3972	35.8519	2.4276	2.3462	35.4418	2.4035	2.3584
	NM	1.7751	25.9447	1.8159	1.7708	25.7156	1.8483	1.747
	MVM	0.4646	7.2283	0.6019	2.4209	36.5599	2.4782	2.1621
	ARM	0.3231	4.8924	0.3827	0.3923	5.8474	0.5299	0.3625
	ANN	0.1692	2.7067	0.2565	0.2316	3.7305	0.5394	0.2921
	REM	**0.149**	**2.2909**	**0.2156**	**0.1863**	**2.6774**	**0.3293**	**0.2096**
20%	MDBM	1.5768	24.1388	1.5828	1.606	24.8715	1.633	1.6747
	NM	1.5472	23.5976	1.5533	1.5252	23.214	1.5537	1.5042
	MVM	0.291	4.4908	0.3219	1.4938	23.1493	1.5229	1.9087
	ARM	0.1119	1.7174	0.1377	0.1856	2.8964	0.2961	0.2418
	ANN	0.1257	2.3816	0.163	0.2866	4.5611	1.0624	0.2024
	REM	**0.1023**	**1.5762**	**0.1262**	**0.1294**	**1.9917**	**0.2296**	**0.1424**
30%	MDBM	0.8668	13.9253	0.8843	0.9315	15.1109	0.9603	0.8968
	NM	1.2654	20.133	1.2774	1.2362	19.7015	1.2581	1.2387
	MVM	0.3038	4.9322	0.3506	0.4767	7.8072	0.5308	1.374
	REM	0.0808	1.2883	0.1039	0.1119	1.8537	0.1981	0.1301
	ARM	**0.1376**	**2.2128**	**0.175**	**0.1484**	**2.4561**	**0.2334**	**0.1538**

Table 4: MEA, MAPE and RMSE comparison between different methods using various malicious density and test-set rate

We divided the working dataset into two sets; the training-set to train the SVR-model, and the test-set to consider their elements as newcomers and thus, to estimate their initial reputation.

Many runs have been performed varying the maliciousness rate and test-set ratio. For instance, Figure 3 depicts a sample of newcomers' estimated reputation values versus the calculated reputation values for two test-sets. As we can see, there is a small variation (less than 0.2 in the worst cases) between the SVR-estimated value and the actual value calculated by feedback ratings.

Besides, we made a comparison between our SVR-model and other methods, using different maliciousness rates and different test-set ratios. The obtained MEA, MAPE, and RMSE are summarized in Table 4. From this table, we can observe that our method (**REM**) gave smaller MAE, MAPE and RMSE values (indicating better accuracy) consistently, with a Percentage error that does not exceed 2.6%. The ANN method gives results relatively close to ours. The ARM method gives also a good result with an error rate that ranges between 1.44 % and 4.5 %. In addition, we observe that the more the data we use for training (i.e. small test-set ratios) the better results we obtain.

7. Related Work

Reputation management has been successfully studied in different computer science domains such as in e-commerce and online information systems (e.g. [45–49]), multi-agent systems (e.g. [50–53]), peer to peer (P2P) and social networks (e.g. [54–56]), and mobile and ad-hoc networking (e.g. [57–59]). For cloud and service-oriented computing systems, many reputation management methods have been proposed (e.g. [1–7, 11, 60–63]). However, little efforts have been dedicated to the study of newcomer reputation estimation.

In fact, most of the proposed reputation approaches do not consider thoroughly this aspect, and a few of them provide default bootstrapping techniques [16, 17, 40, 41, 64], i.e. they assign a default constant reputation score, such as a low (0), a neutral (0.5), a high (the maximum) or no reputation value, to all newcomers. For instance, Zacharia *et al.* [40] give the minimum possible trust value, and [41] assigns a neutral value (0.5). However, in such situation, newcomers may never have a chance to get selected. Even in the

case of assigning high trust values, the problem of "whitewashers" (i.e. malicious participants that leave the system and come back with new identities) could raise.

The framework proposed by Jin-Dian *et al.* [65] assigns the provider reputation to its newly posted services. The authors suggest assessing the reputation of the provider based on its past experiences. However, the problem appears if the provider is a newcomer in the recommendation system.

Feldman and Chuang [18] propose a solution for bootstrapping the reputation of newcomer services based on its probability of deceiving. This probability is computed by collecting all transaction information of the newcomer's first-time interaction. This approach is community-based, and newcomer reputation is adjusted to the reputation of others. However, initial reputation scores are still not fair and they do not reflect the actual reputation of newcomers.

Malik and Bouguettaya [14] propose two bootstrapping techniques for establishing the reputation of newcomer web services. The first is an adaptive technique that assigns the initial reputation value based on the rate of maliciousness in the system. The second approach assigns a default reputation score to a newcomer service, where the initial reputation is purchased from the community provider. Or, the community requests some evaluators (elder service with high reputation) to evaluate the newcomer service in a short period of time. In the first technique, the reputation of a specific web service is related to the maliciousness rate in the community, which seems penalizing or rewarding based on a factor that is unrelated to the service itself. In the second technique, the contribution and the impact of requesters on the reputation of web services are very high, which raises the problem of the trust of evaluators themselves.

Huang *et al.* [13] propose an equitable trustworthy mechanism that enables new services to startup and grow in an ecosystem environment. The mechanism distinguishes between novice and mature services during service recommendation. The approach considers two trust bootstrapping strategies: i) default strategy where they assign to the newcomer a default initial trust value, and ii) an adaptive bootstrapping strategy where they assign to the newcomer the average trust value in the system. The first strategy does not provide a solution for the cold-start problem and for the whitewashing problem in the case of assigning a high value. Moreover, the second technique assigns the average trust in the system to newcomer services, which is not always an accurate solution (e.g. the case where the average is high and

the service is bad or the inverse).

Wu *et al.* [16] introduce a neural network based approach for bootstrapping the reputation of web services. The approach builds a model that learns possible correlations between features and performances of existing services using Artificial Neural Networks. Then, it generalizes findings to establish tentative reputation values when it evaluates new and unknown services. This approach depends on features that are gathered from service providers by filling a specific form without taking into consideration the whitewashing cases.

The main differences between our solution and the previous solutions can be summarized as follows. First, instead of assigning the same reputation value to all newcomer web services, we propose to estimate the initial reputation value of a newcomer service based on its provider's reputation, reputation values of its similar long-standing services, and its initial QoS values. When all the previous values are not present, we make an estimation using a Support vector regression model built from the reputation and QoS values of long-standing services. In addition, we propose a solution to overcome the whitewashing problem based on similarity of the newcomer service with registered services that have left the system.

8. Conclusion

In this work, we proposed a method that estimates reputation values of newcomer web services. Initial reputation values assigned to newcomer web services have an impact on the performance of web service recommendation systems. Our proposed method uses (i) provider reputation values, (ii) neighbor services' reputation values, and (iii) SVR-based reputation estimation model, trained by QoS and reputation values of long-standing web services, to correctly estimate reputation values of newcomer web services. This method addresses both the cold start and the whitewashing problems often encountered in online recommendation systems.

We conducted several experiments on a real Web service dataset to evaluate the efficiency of the proposed method. Through experimental evidence, we showed that the proposed method outperforms existing methods and may be safely applied to estimate the reputation of newcomers in Web service recommendation systems.

Our future research work includes the proposition of a complete service recommendation system with a unified reputation management and security

model for both single, composite, and community-based Web services. These models target more online attacks such as the request drop, denial of service, outage, and eavesdropping attacks.

References

[1] Y. Wang, C. Guo, T. Li, Q. Xu, Secure Two-Party Computation in Social Cloud Based on Reputation, in: Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on, IEEE, 242–245, 2015.

[2] Z. Maamar, G. Costantino, M. Petrocchi, F. Martinelli, Business Reputation of Social Networks of Web Services, Procedia Computer Science 56 (2015) 18 – 25, ISSN 1877-0509, the 10th International Conference on Future Networks and Communications (FNC 2015) / The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2015) Affiliated Workshops.

[3] F. Moyano, C. Fernandez-Gago, J. Lopez, A Model-driven Approach for Engineering Trust and Reputation into Software Services, Journal of Network and Computer Applications (2016) –ISSN 1084-8045.

[4] M. Mehdi, N. Bouguila, J. Bentahar, Trust and Reputation of Web services Through QoS Correlation Lens, IEEE Transactions on Services Computing PP (99) (2015) 1–1.

[5] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, F. Yang, Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems, IEEE Transactions on Services Computing 8 (5) (2015) 755–767, ISSN 1939-1374.

[6] T. H. Noor, Q. Z. Sheng, S. Zeadally, J. Yu, Trust management of services in cloud environments: Obstacles and solutions, ACM Comp. Surveys 46 (1) (2013) 12.

[7] F. Hendrikx, K. Bubendorfer, R. Chard, Reputation systems: A survey and taxonomy, Journal of Parallel and Distributed Computing 75 (2015) 184–197.

[8] O. A. Wahab, J. Bentahar, H. Otrok, A. Mourad, A survey on trust and reputation models for Web services: Single, composite, and communities, Decision Support Systems 74 (2015) 121–134.

[9] Z. M. Aljazzaf, Trust-based service selection, Ph.D. thesis, The University of Western Ontario, 2011.

[10] H. T. Nguyen, J. Yang, W. Zhao, Bootstrapping trust and reputation for Web services, in: Commerce and Enterprise Computing (CEC), 2012 IEEE 14th International Conference on, IEEE, 41–48, 2012.

[11] F. G. Mármol, M. Q. Kuhnen, Reputation-based Web service orchestration in cloud computing: A survey, Concurrency and Computation: Practice and Experience 27 (9) (2015) 2390–2412.

[12] J. B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in: The adaptive web, Springer, 291–324, 2007.

[13] K. Huang, Y. Liu, S. Nepal, Y. Fan, S. Chen, W. Tan, A Novel Equitable Trustworthy Mechanism for Service Recommendation in the Evolving Service Ecosystem, in: Service-Oriented Computing, Springer, 510–517, 2014.

[14] Z. Malik, A. Bouguettaya, Reputation bootstrapping for trust establishment among web services, Internet Computing, IEEE 13 (1) (2009) 40–47.

[15] J. R. Douceur, The sybil attack, in: Peer-to-peer Systems, Springer, 251–260, 2002.

[16] Q. Wu, Q. Zhu, P. Li, A neural network based reputation bootstrapping approach for service selection, Enterprise Information Systems 9 (7) (2015) 768–784.

[17] M. Chen, L. He, X. Cai, W. Xia, Trust evaluation model for composite service based on subjective logic, in: Proc. of IIHMSP'08, IEEE, 1482–1485, 2008.

[18] M. Feldman, J. Chuang, The evolution of cooperation under cheap pseudonyms, in: Proc. of CEC'05, IEEE, 284–291, 2005.

[19] O. Tibermacine, C. Tibermacine, F. Cherif, Regression-Based Bootstrapping of Web Service Reputation Measurement, in: Web Services (ICWS), 2015 IEEE International Conference on, IEEE, 377–384, 2015.

[20] Z. Malik, A. Bouguettaya, Rateweb: Reputation assessment for trust establishment among web services, VLDB Journal 18 (4) (2009) 885–911.

[21] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and computing 14 (3) (2004) 199–222.

[22] M. Awad, R. Khanna, Support Vector Regression, in: Efficient Learning Machines, Springer, 67–80, 2015.

[23] M. H. Hasan, J. Jaafar, M. F. Hassan, Monitoring web services quality of service: a literature review, Artificial Intelligence Review 42 (4) (2014) 835–850.

[24] O. Tibermacine, C. Tibermacine, F. Cherif, A Practical Approach to the Measurement of Similarity between WSDL-based Web Services, Revue des Nouvelles Technologies de l'Information 6th French-speaking Conference on Software Architectures, RNTI-L-7 (2014) 03–18.

[25] M. Garriga, A. Flores, C. Mateos, A. Zunino, A. Cechich, Service selection based on a practical interface assessment scheme, International Journal of Web and Grid Services 9 (4) (2013) 369–393, ISSN 1741-1106.

[26] N. Kokash, A comparison of web service interface similarity measures, Frontiers in Artificial Intelligence and Applications 142 (2006) 220, ISSN 0922-6389.

[27] O. Tibermacine, C. Tibermacine, F. Cherif, WSSim: a Tool for the Measurement of Web Service Interface Similarity, in: French-speaking Conference on Software Architectures (CAL'13), 2013.

[28] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, Recommender Systems: An Introduction–Cambridge University Press, New York, 2010.–352 P .

[29] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Transactions on Information Systems (TOIS) 22 (1) (2004) 143–177.

[30] Z. Zheng, H. Ma, M. R. Lyu, I. King, Collaborative web service qos prediction via neighborhood integrated matrix factorization, Services Computing, IEEE Transactions on 6 (3) (2013) 289–299.

[31] H. Ma, I. King, M. R. Lyu, Effective missing data prediction for collaborative filtering, in: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 39–46, 2007.

[32] V. Vapnik, The nature of statistical learning theory, Springer Science & Business Media, 2013.

[33] W. Zhao, T. Tao, E. Zio, System reliability prediction by support vector regression with analytic selection and genetic algorithm parameters selection, Applied Soft Computing 30 (2015) 792–802.

[34] R. Chen, C.-Y. Liang, W.-C. Hong, D.-X. Gu, Forecasting holiday daily tourist flow based on seasonal support vector regression with adaptive genetic algorithm, Applied Soft Computing 26 (2015) 435–443.

[35] J. Hu, J. Qi, Y. Peng, Q. Ren, Predicting electrical evoked potential in optic nerve visual prostheses by using support vector regression and case-based prediction, Information Sciences 290 (2015) 7–21.

[36] A. Kavousi-Fard, H. Samet, F. Marzbani, A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting, Expert systems with applications 41 (13) (2014) 6047–6056.

[37] L. Ghelardoni, A. Ghio, D. Anguita, Energy load forecasting using empirical mode decomposition and support vector regression, Smart Grid, IEEE Transactions on 4 (1) (2013) 549–556.

[38] Z. Zheng, Y. Zhang, M. R. Lyu, Distributed qos evaluation for real-world web services, in: Proc. of ICWS'10, IEEE, 83–90, 2010.

[39] E. Al-Masri, Q. H. Mahmoud, Qos-based discovery and ranking of web services, in: Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on, IEEE, 529–534, 2007.

[40] G. Zacharia, A. Moukas, P. Maes, Collaborative reputation mechanisms for electronic marketplaces, Decision Support Systems 29 (4) (2000) 371–388.

[41] X. Wang, K. Govindan, P. Mohapatra, Provenance-based information trustworthiness evaluation in multi-hop networks, in: Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, IEEE, 1–5, 2010.

[42] N. Limam, R. Boutaba, Assessing software service quality and trustworthiness at selection time, IEEE Transactions on Software Engineering, 36 (4) (2010) 559–574.

[43] A. Whitby, A. Jøsang, J. Indulska, Filtering out unfair ratings in bayesian reputation systems, in: Proc. 7th Int. Workshop on Trust in Agent Societies, vol. 6, 2004.

[44] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 27.

[45] B. Tian, J. Han, K. Liu, Closed-Loop Feedback Computation Model of Dynamical Reputation Based on the Local Trust Evaluation in Business-to-Consumer E-Commerce, Information 7 (1) (2016) 4.

[46] D. Isherwood, M. Coetzee, Trust CV: Reputation-based trust for collectivist digital business ecosystems, in: Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on, IEEE, 420–424, 2014.

[47] M.-H. Peetz, M. de Rijke, R. Kaptein, Estimating Reputation Polarity on Microblog Posts, Information Processing & Management 52 (2) (2016) 193 – 216, ISSN 0306-4573.

[48] J. Hu, Y. Zhang, Research patterns and trends of Recommendation System in China using co-word analysis, Information Processing & Management 51 (4) (2015) 329 – 339, ISSN 0306-4573.

[49] A. J. Bidgoly, B. T. Ladani, Benchmarking reputation systems: A quantitative verification approach, Computers in Human Behavior 57 (2016) 274–291.

[50] A. Comi, L. Fotia, F. Messina, G. Pappalardo, D. Rosaci, G. M. Sarné, A Distributed Reputation-Based Framework to Support Communication Resources Sharing, in: Intelligent Distributed Computing IX, Springer, 211–221, 2016.

[51] L. Barakat, S. Mahmoud, P. Taylor, N. Griffiths, S. Miles, Reputation-based provider incentivisation for provenance provision, in: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), 2016.

[52] W. Itani, C. Ghali, A. Kayssi, A. Chehab, Reputation as a Service: A System for Ranking Service Providers in Cloud Systems, in: Security, Privacy and Trust in Cloud Systems, Springer, 375–406, 2014.

[53] E. Majd, V. Balakrishnan, A trust model for recommender agent systems, Soft Computing (2016) 1–17.

[54] H. Zhao, X. Li, VectorTrust: trust vector aggregation scheme for trust management in peer-to-peer networks, The Journal of Supercomputing 64 (3) (2013) 805–829.

[55] A. Louati, J. El Haddad, S. Pinson, A Multi-Agent Approach for Trust-based Service Discovery and Selection in Social Networks, Scalable Computing: Practice and Experience 16 (4) (2016) 381–402.

[56] B. Zhang, Q. Song, T. Yang, Z. Zheng, H. Zhang, A Fuzzy Collusive Attack Detection Mechanism for Reputation Aggregation in Mobile Social Networks: A Trust Relationship Based Perspective, Mobile Information Systems 2016.

[57] J.-H. Cho, A. Swami, R. Chen, A survey on trust management for mobile ad hoc networks, Communications Surveys & Tutorials, IEEE 13 (4) (2011) 562–583.

[58] S. Sutariya, P. Modi, A Review of Different Reputation Schemes to Thwart the Misbehaving Nodes in Mobile Ad Hoc Network., International Journal of Computer Science & Information Technologies 5 (3).

[59] W. Zheng, L. Jin, A Consumer Decision-Making Model in M-Commerce: The Role of Reputation Systems in Mobile App Purchases, Information Resources Management Journal (IRMJ) 29 (2) (2016) 37–58.

[60] Z. Liu, J. Ma, Z. Jiang, Y. Miao, C. Gao, IRLT: Integrating Reputation and Local Trust for Trustworthy Service Recommendation in Service-Oriented Social Networks, PloS one 11 (3) (2016) e0151438.

[61] J. Yao, W. Tan, S. Nepal, S. Chen, J. Zhang, D. D. Roure, C. Goble, ReputationNet: Reputation-Based Service Recommendation for e-Science, IEEE Transactions on Services Computing 8 (3) (2015) 439–452, ISSN 1939-1374.

[62] Z. Yan, X. Li, R. Kantola, Controlling Cloud Data Access Based on Reputation, Mobile Networks and Applications 20 (6) (2015) 828–839.

[63] C. Zhu, H. Nicanfar, V. Leung, L. T. Yang, An authenticated trust and reputation calculation and management system for cloud and sensor networks integration, Information Forensics and Security, IEEE Transactions on 10 (1) (2015) 118–131.

[64] T. D. Huynh, N. R. Jennings, N. R. Shadbolt, Certified reputation: how an agent can trust a stranger, in: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, ACM, 1217–1224, 2006.

[65] S. Jin-dian, G. He-qing, G. Yin, An adaptive trust model of Web services, Wuhan University Journal of Natural Sciences 10 (1) (2005) 21–25.